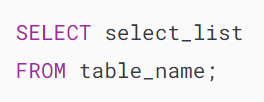
# **MySQL SELECT FROM**

**Summary**: in this tutorial, you will learn how to use the basic form of the MySQL SELECT FROM statement to query data from a table.

## **Introduction to MySQL SELECT FROM statement**

The SELECT statement allows you to select data from one or more tables. To write a SELECT statement in MySQL, you use this syntax:

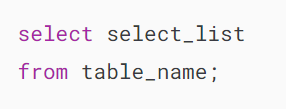


In this syntax:

* First, specify one or more columns from which you want to select data after the SELECT keyword. If the select\_list has multiple columns, you need to separate them by a comma (,).
* Second, specify the name of the table from which you want to select data after the FROM keyword.

The semicolon (;) is optional, which denotes the end of a statement. If you have two or more statements, you need to use the semicolon(;) to separate them so that MySQL will execute each statement individually.

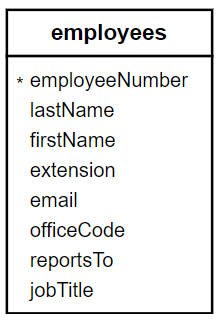
The SELECT and FROM are the keywords. By convention, you write the SQL keywords in uppercase. However, it’s not mandatory. Because SQL is case-insensitive, you can write the SQL statement in lowercase, uppercase, etc. For example:



When executing the SELECT statement, MySQL evaluates the FROM clause before the SELECT clause:

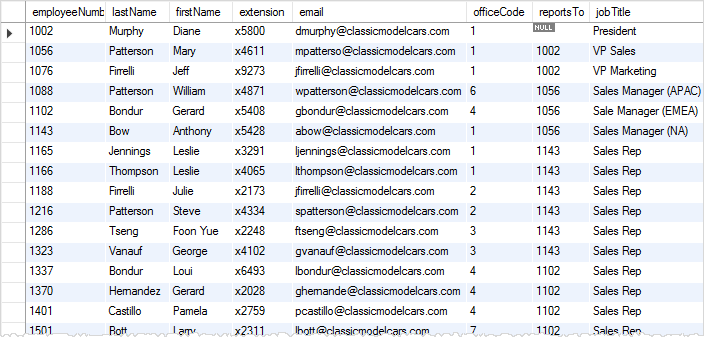
## **MySQL SELECT FROM statement examples**

We’ll use the employees table in the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/) for the following examples:



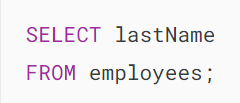
The employees table has eight columns: employeeNumber, lastName, firstName, extension, email, officeCode, reportsTo, and jobTitle.

The table also has many rows as shown in the following picture:



### **1) Using the SELECT FROM statement to retrieve data from a single column example**

The following example uses the SELECT FROM statement to retrieve the last names of all employees:



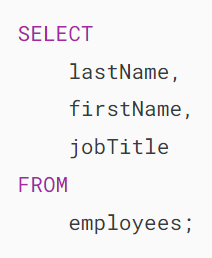
Here’s the partial output:



The result of a SELECT statement is called a **result set** as it’s a set of rows that results from the query.

### **2) Using the SELECT FROM statement to query data from multiple columns example**

The following example uses the SELECT FROM statement to get the first name, last name, and job title of employees:

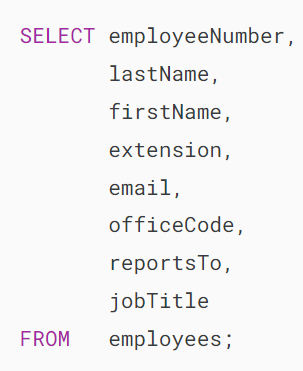


Even though the employees table has many columns, the SELECT statement returns data of three columns lastName, firstName, and jobTitle specified in the SELECT clause:

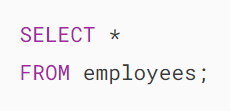
### 

### **3) Using the SELECT FROM statement to retrieve data from all columns example**

If you want to select data from all the columns of the employees table, you can specify all the column names in the SELECT clause like this:



Alternatively, you can use the asterisk (\*) which is the shorthand for all columns. For example:



The query returns data from all the columns of the employees table.

The SELECT \* is often called “select star” or “select all” since it selects data from all columns of the table. In practice, you should use the SELECT \* for the ad-hoc queries only.

If you embed the SELECT statement in the code such as [PHP](https://www.mysqltutorial.org/php-mysql/), [Java](https://www.mysqltutorial.org/mysql-jdbc-tutorial/), [Python](https://www.mysqltutorial.org/python-mysql/), and [Node.js](https://www.mysqltutorial.org/mysql-nodejs/), you should explicitly specify the columns from which you want to select data.

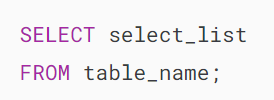
## **Summary**

* Use the SELECT FROM statement to select data from a table.
* Use the SELECT \* to select data from all columns of a table.

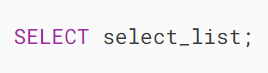
# **MySQL SELECT**

**Summary**: in this tutorial, you’ll learn how to use the MySQL SELECT statement without referencing any table.

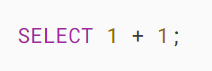
Typically, you use a SELECT statement to select data from a table in the database:



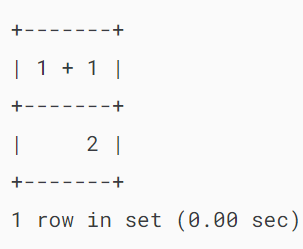
In MySQL, the SELECT statement doesn’t require the FROM clause. It means that you can have a SELECT statement without the FROM clause like this:



The following example uses the SELECT statement to perform a simple calculation:



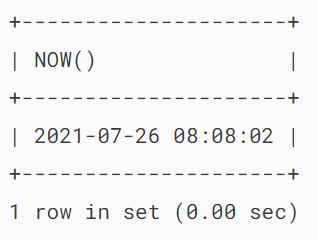
Output:



MySQL has many built-in functions like [string functions](https://www.mysqltutorial.org/mysql-string-functions/), [date functions](https://www.mysqltutorial.org/mysql-date-functions/), and [math functions](https://www.mysqltutorial.org/mysql-math-functions/). You can use the SELECT statement to execute one of these functions.

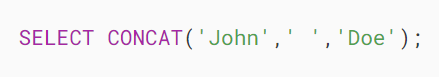
For example, the following statement uses the NOW() function in the SELECT statement to return the current date and time of the server where MySQL server is running:



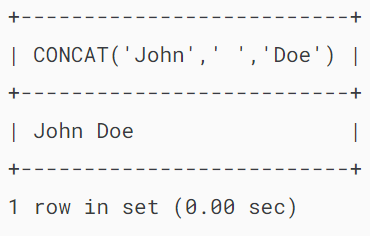


The NOW() function doesn’t have any parameters. To call it, you place the parentheses () after the function name.

If a function has parameters, you need to pass arguments into it. For example, to concatenate multiple strings into a single string, you can use the [CONCAT()](https://www.mysqltutorial.org/mysql-string-functions/mysql-concat/) function:



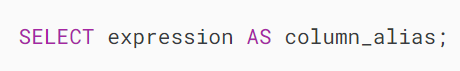
Output:



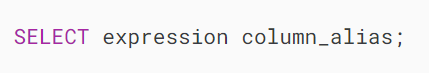
The CONCAT() function accepts one or more strings and concatenates them into a single string.

## **Column alias**

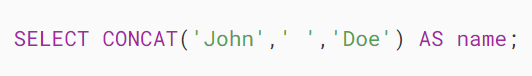
By default, MySQL uses the expression specified in the SELECT clause as the column name of the result set. To change a column name of the result set, you can use a column alias:



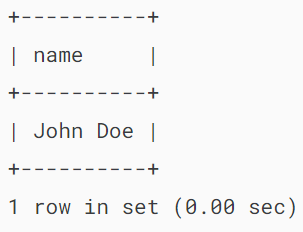
To assign an alias to a column, you place the AS keyword after the expression followed by a column alias. The AS keyword is optional, so you can skip it like this:



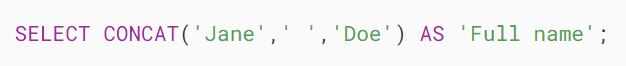
For example:



Output:



If the column alias contains spaces, you need to place it inside quotes like this:



Output:

## 

## **Summary**

* MySQL SELECT statement doesn’t require the FROM clause.
* Assign an alias to a column to make it more readable.

# **MySQL ORDER BY**

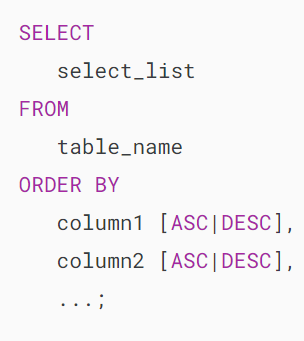
**Summary**: in this tutorial, you will learn how to sort the rows in a result set using the MySQL ORDER BY clause.

## **Introduction to the MySQL ORDER BY clause**

When you use the [SELECT](https://www.mysqltutorial.org/mysql-basics/mysql-select-from/) statement to query data from a table, the order of rows in the result set is unspecified.

To sort the rows in the result set, you add the ORDER BY clause to the SELECT statement.

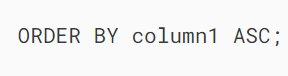
The following illustrates the syntax of the ORDER BY clause:



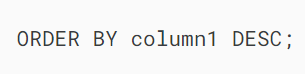
In this syntax, you specify the one or more columns that you want to sort after the ORDER BY clause.

The ASC stands for ascending and the DESC stands for descending. You use ASC to sort the result set in ascending order and DESC to sort the result set in descending order.

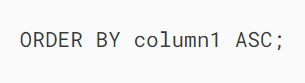
This ORDER BY clause sorts the result set by the values in the column1 in ascending order:



And this ORDER BY clause sorts the result set by the values in the column1 in descending order:



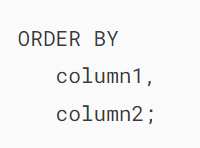
By default, the ORDER BY clause uses ASC if you don’t explicitly specify any option. Therefore, the following ORDER BY clauses are equivalent:



and

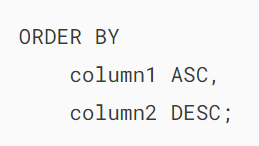


If you want to sort the result set by multiple columns, you specify a comma-separated list of columns in the ORDER BY clause:



In this case, the ORDER BY clause sorts the result set by column1 in ascending order first and sorts the sorted result set by column2 in ascending order.

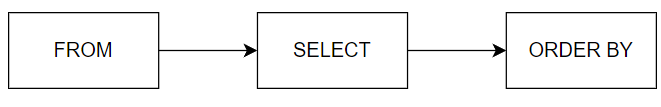
It is possible to sort the result set by a column in ascending order and then by another column in descending order:



In this case, the ORDER BY clause:

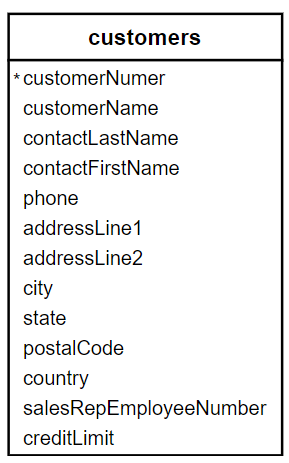
* First, sort the result set by the values in the column1 in ascending order.
* Then, sort the sorted result set by the values in the column2  in descending order. Note that the order of values in the column1 will not change in this step, only the order of values in the column2 changes.

When executing the SELECT statement with an ORDER BY clause, MySQL always evaluates the ORDER BY clause after the FROM and SELECT clauses:



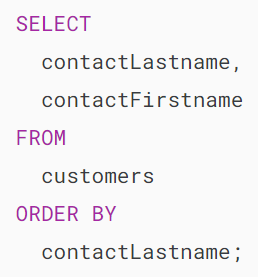
## **MySQL ORDER BY examples**

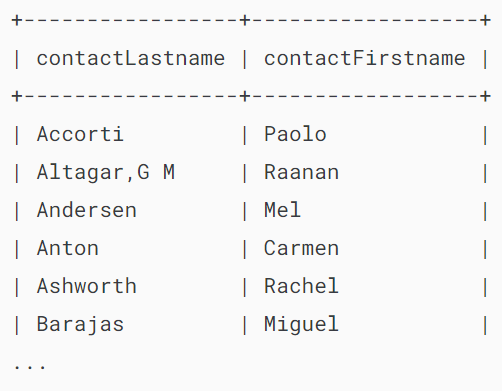
We’ll use the customers table from the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/) for the demonstration:



### **1) Using ORDER BY clause to sort the result set by one column example**

The following query uses the ORDER BY clause to sort the customers by their last names in ascending order.





If you want to sort customers by the last name in descending order, you use the DESC after the contactLastname column in the ORDER BY clause as shown in the following query:



### 

### **2) Using the ORDER BY clause to sort the result set by multiple columns example**

If you want to sort the customers by the last name in descending order and then by the first name in ascending order, you specify both  DESC and ASC in these respective columns as follows:



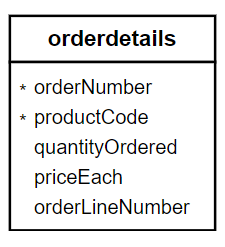
Output:



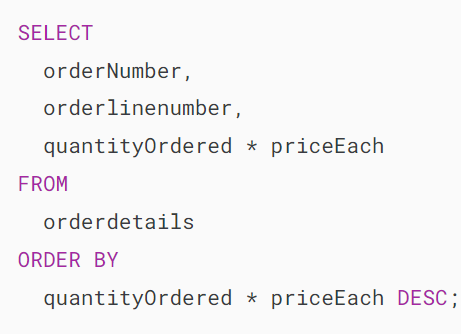
In this example, the ORDER BY  clause sorts the result set by the last name in descending order first and then sorts the sorted result set by the first name in ascending order to make the final result set.

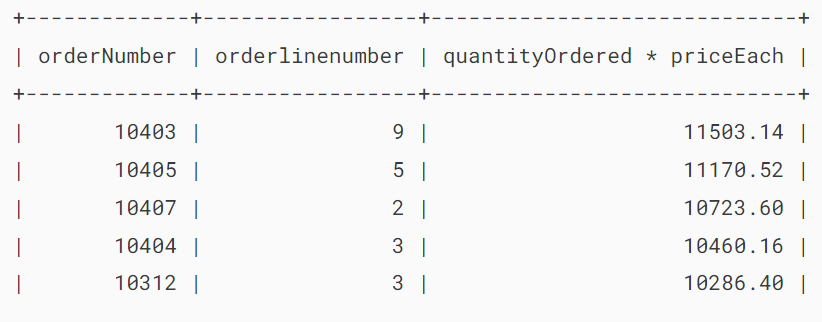
### **3) Using the ORDER BY clause to sort a result set by an expression example**

See the following orderdetails table from the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/):

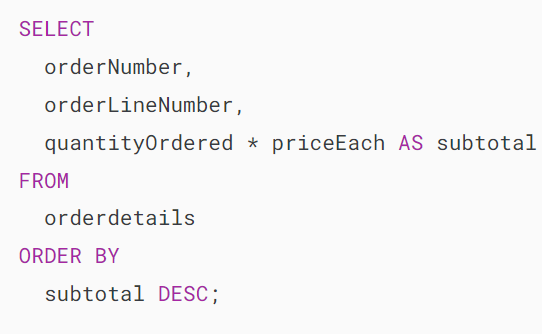


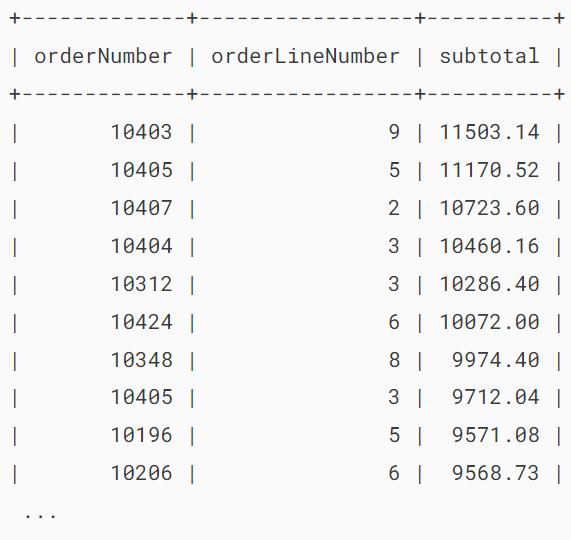
The following query selects the order line items from the orderdetails table. It calculates the subtotal for each line item and sorts the result set based on the subtotal.





To make the query more readable, you can assign a [column alias](https://www.mysqltutorial.org/mysql-basics/mysql-alias/) to the expression in the SELECT clause and use the column alias in the ORDER BY clause as shown in the following query:





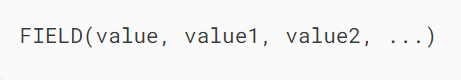
In this example, we use subtotal as the [column alias](https://www.mysqltutorial.org/mysql-basics/mysql-alias/) for the expression quantityOrdered \* priceEach and sort the result set by the subtotal alias.

Since MySQL evaluates the SELECT clause before the ORDER BY clause, you can use the column alias specified in the SELECT clause in the ORDER BY clause.

## **Using MySQL ORDER BY clause to sort data using a custom list**

The FIELD() function returns the index (position) of a value within a list of values.

Here’s the syntax of the FIELD() function:



In this syntax:

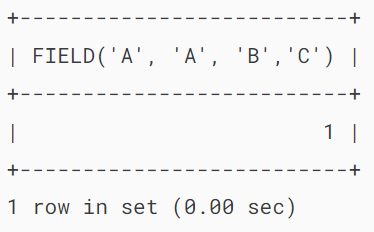
* value: The value for which you want to find the position.
* value1, value2, ...: A list of values against which you want to compare the specified value.

The FIELD() function returns the position of the value in the list of values value1, value2, and so on.

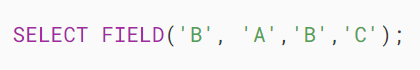
If the value is not found in the list, the FIELD() function returns 0.

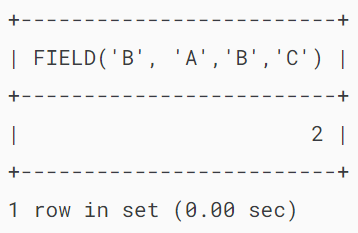
For example, the following query returns 1 because the position of the string 'A' is the first position on the list 'A', 'B', and 'C':





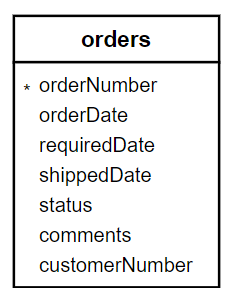
And the following example returns 2 because 'B' has the second position in the list:





Let’s take a more practical example.

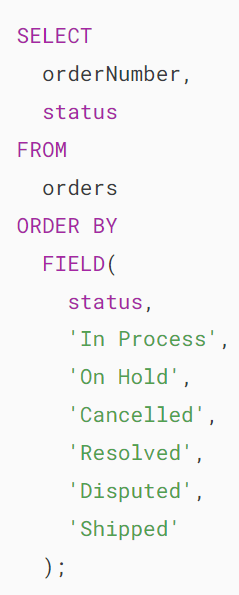
See the following orders table from the sample database:



Suppose that you want to sort the sales orders based on their statuses in the following order:

* In Process
* On Hold
* Canceled
* Resolved
* Disputed
* Shipped

To do this, you can use the FIELD() function to map each order status to a number and sort the result by the result of the FIELD() function:

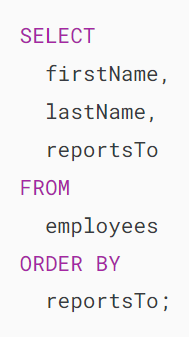


## 

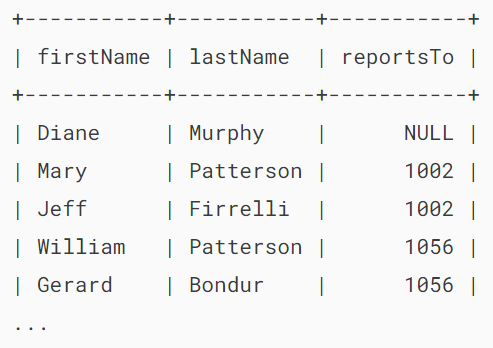
## **MySQL ORDER BY and NULL**

In MySQL, NULL comes before non-NULL values. Therefore, when you the ORDER BY clause with the ASC option, NULLs appear first in the result set.

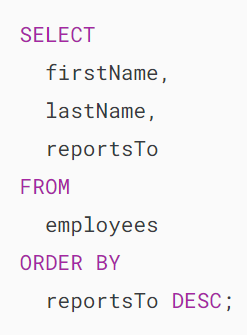
For example, the following query uses the ORDER BY clause to sort employees by values in the reportsTo column:



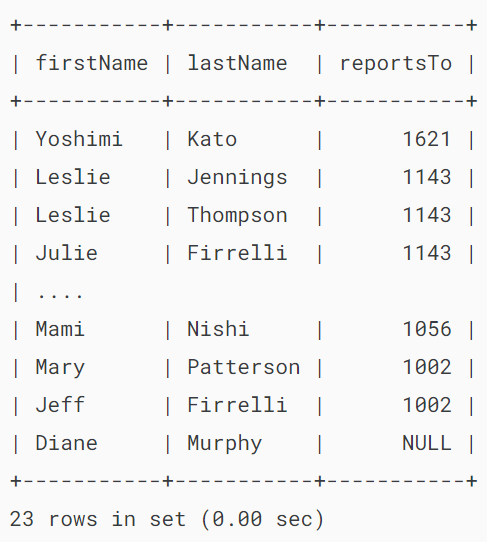
Output:



However, if you use the ORDER BY with the DESC option, NULLs will appear last in the result set. For example:



Output:



## **Summary**

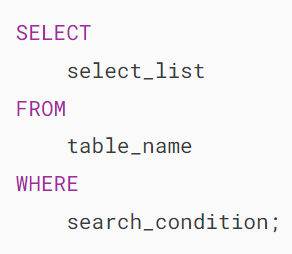
* Use the ORDER BY clause to sort the result set by one or more columns.
* Use the ASC option to sort the result set in ascending order and the DESC option to sort the result set in descending order.
* The ORDER BY clause is evaluated after the FROM and SELECT clauses.
* In MySQL, NULL is lower than non-NULL values

# **MySQL WHERE**

**Summary:**in this tutorial, you will learn how to use the MySQL WHERE clause in the SELECT statement to filter rows from the result set.

## **Introduction to MySQL WHERE clause**

The WHERE clause allows you to specify a search condition for the rows returned by a query. The following shows the syntax of the WHERE clause:



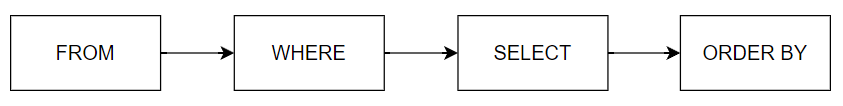
The search\_condition is a combination of one or more expressions using the logical operator [AND](https://www.mysqltutorial.org/mysql-basics/mysql-and/), [OR](https://www.mysqltutorial.org/mysql-basics/mysql-or/) and NOT.

In MySQL, a predicate is a [Boolean](https://www.mysqltutorial.org/mysql-basics/mysql-boolean/) expression that evaluates to TRUE, FALSE, or UNKNOWN.

The SELECT statement will include any row that satisfies the search\_condition in the result set.

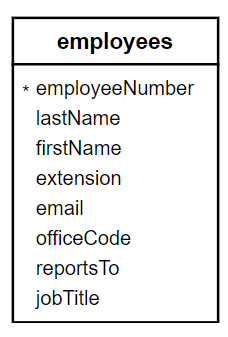
Besides the SELECT statement, you can use the WHERE clause in the [UPDATE](https://www.mysqltutorial.org/mysql-basics/mysql-update/) or [DELETE](https://www.mysqltutorial.org/mysql-basics/mysql-delete/) statement to specify which rows to update or delete.

When executing a SELECT statement with a WHERE clause, MySQL evaluates the WHERE clause after the FROM clause and before the SELECT and ORDER BY clauses:



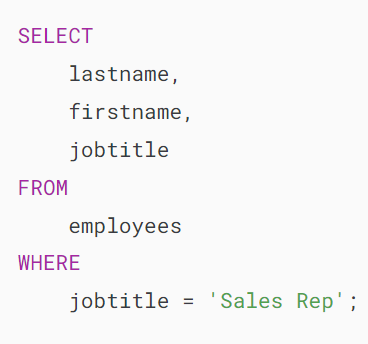
## **MySQL WHERE clause examples**

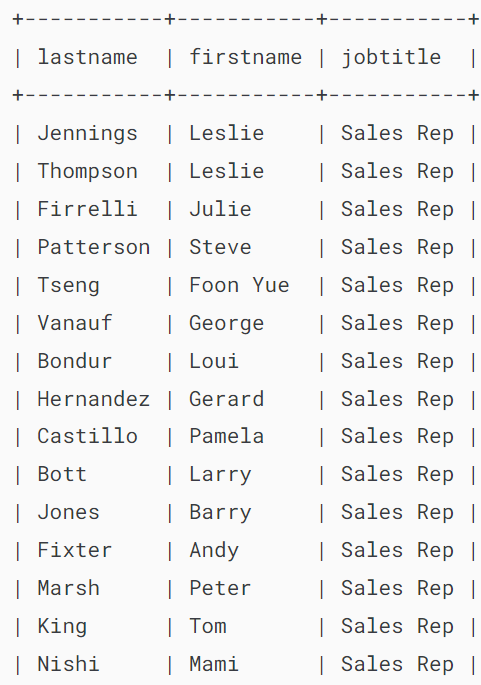
We’ll use the employees table from the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/) for the demonstration:



### **1) Using the WHERE clause with equality operator example**

The following query uses the WHERE clause to find all employees whose job titles are Sales Rep:



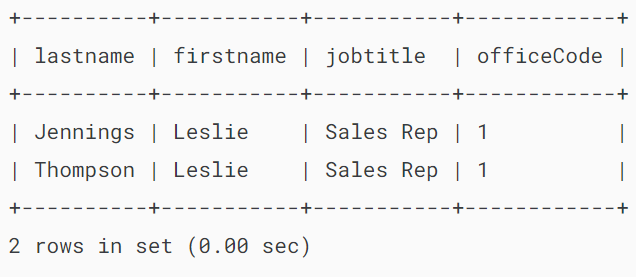


In this example, the SELECT statement examines all rows of the employees table and selects only rows whose values are in the jobTitle column are Sales Rep.

### **2) Using the WHERE clause with the AND operator**

The following example uses the WHERE clause to find employees whose job titles are Sales Rep and office codes are 1:





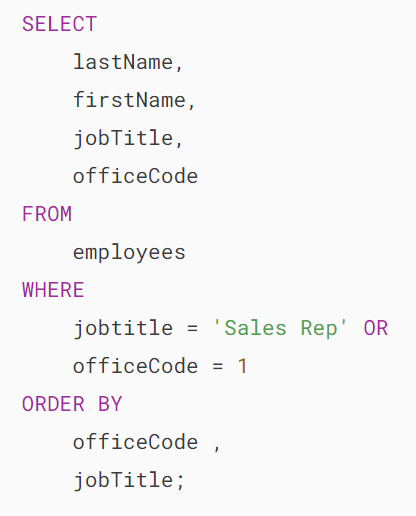
In this example, the expression in the WHERE clause uses the [AND](https://www.mysqltutorial.org/mysql-basics/mysql-and/) operator to combine two conditions:

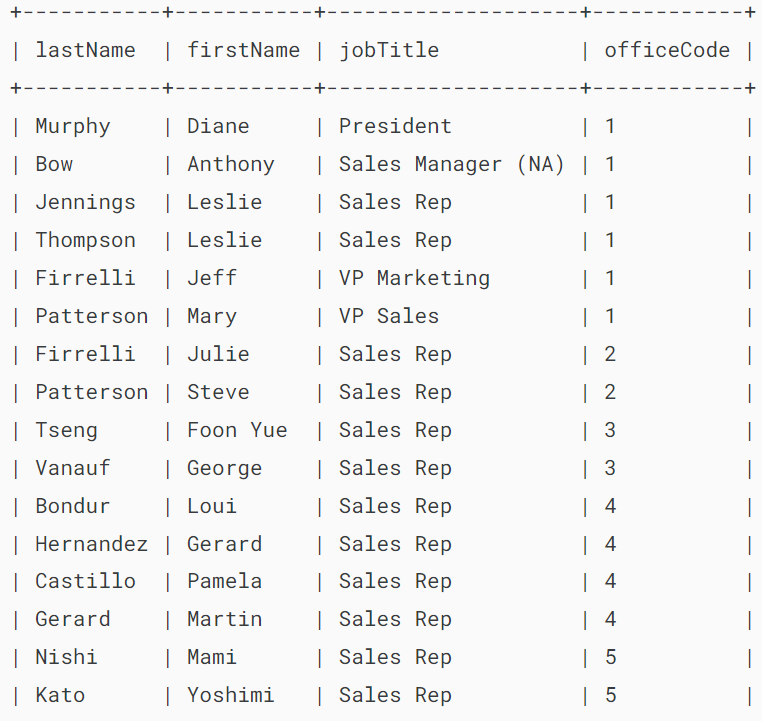


The AND operator evaluates to TRUE only if both expressions are evaluated to TRUE. Therefore, the query returns rows whose values in the jobTitle column is Sales Rep and officeCode is 1.

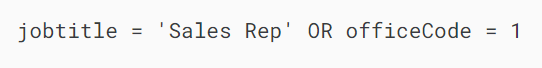
### **3) Using MySQL WHERE clause with OR operator**

This query finds employees whose job title is Sales Rep or employees who locate the office with office code 1:





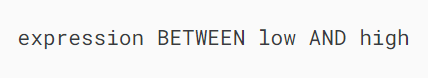
The [OR](https://www.mysqltutorial.org/mysql-basics/mysql-or/) operator evaluates to TRUE only if one of the expressions evaluates to TRUE:



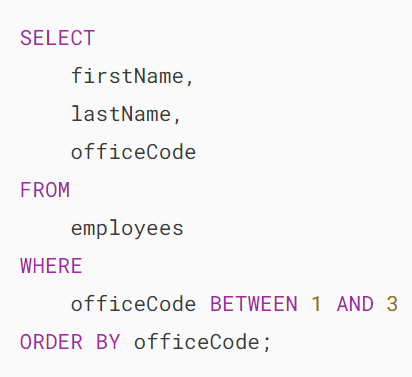
Therefore, the query returns any employee who has the job title Sales Rep or office code 1.

### **4) Using the WHERE clause with the BETWEEN operator example**

The [BETWEEN](https://www.mysqltutorial.org/mysql-between) operator returns TRUE if a value is in a range of values:



The following query finds employees who are located in offices whose office code is from 1 to 3:



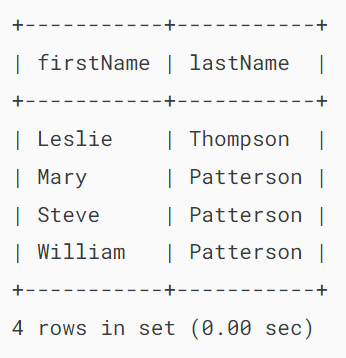
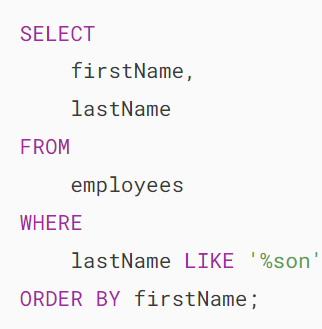
### 

### **5) Using the WHERE clause with the LIKE operator example**

The [LIKE](https://www.mysqltutorial.org/mysql-basics/mysql-like/) operator evaluates to TRUE if a value matches a specified pattern.

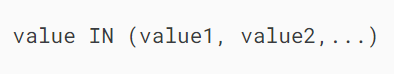
To form a pattern, you use the % and \_ wildcards. The % wildcard matches any string of zero or more characters while the \_ wildcard matches any single character.

The following query finds the employees whose last names end with the string 'son':

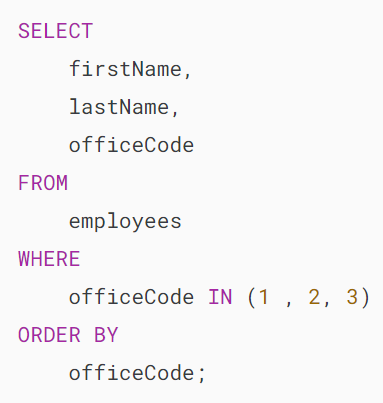


### **6) Using the WHERE clause with the IN operator example**

The [IN](https://www.mysqltutorial.org/mysql-basics/mysql-in/) operator returns TRUE if a value matches any value in a list.



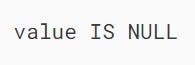
The following example uses the WHERE clause with the IN operator to find employees who are located in the offices with the codes 1, 2, and 3:



### 

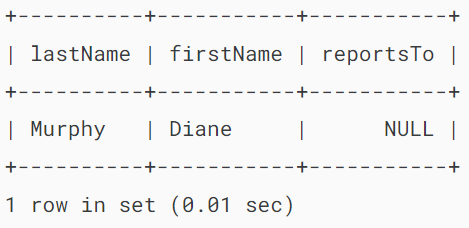
### **7) Using MySQL WHERE clause with the IS NULL operator**

To check if a value is [NULL](https://www.mysqltutorial.org/mysql-basics/mysql-null/) or not, you use the [IS NULL](https://www.mysqltutorial.org/mysql-basics/mysql-is-null/) operator, not the equal operator (=). The IS NULL operator returns TRUE if a value is NULL.



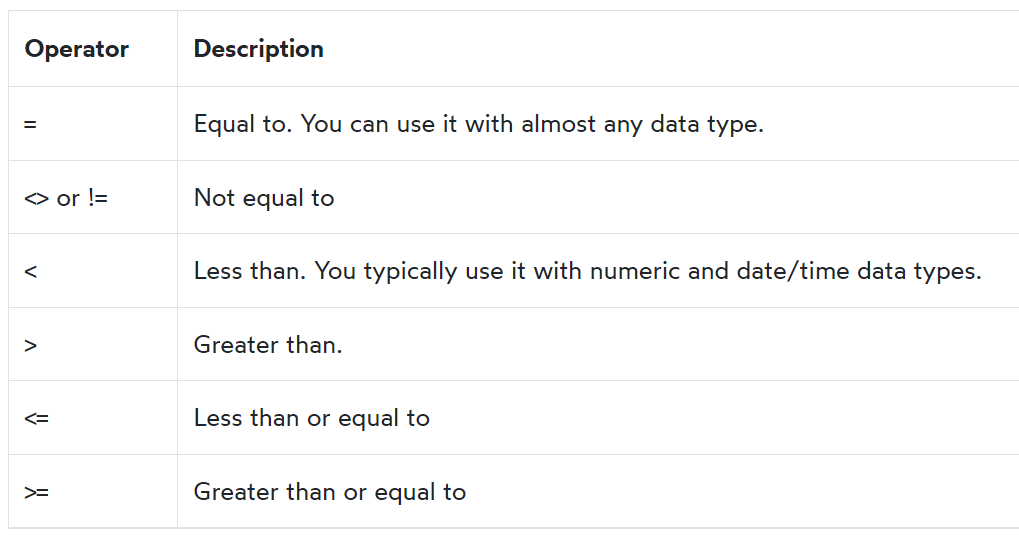
In the database world, NULL is a marker that indicates that a value is missing or unknown. NULL is not equivalent to the number 0 or an empty string.

The following statement uses the WHERE clause with the IS NULL operator to get the rows with the values in the reportsTo column are NULL:

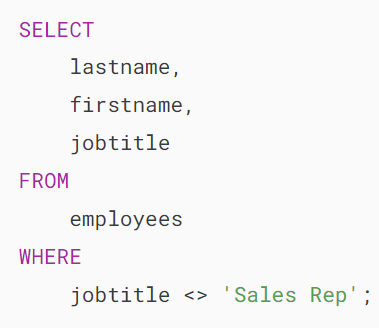


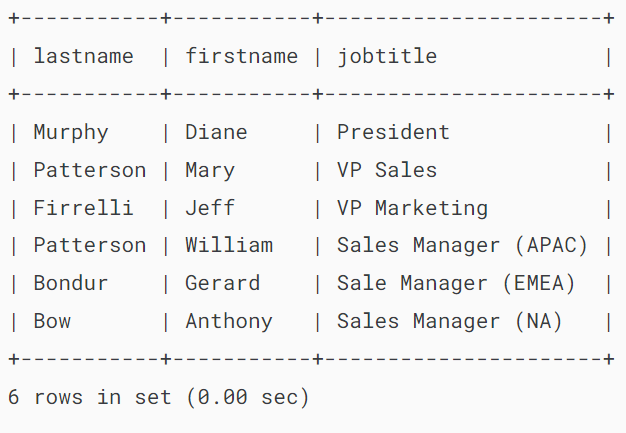
**8) Using MySQL WHERE clause with comparison operators**

The following table shows the comparison operators that you can use to form the expression in the WHERE clause.

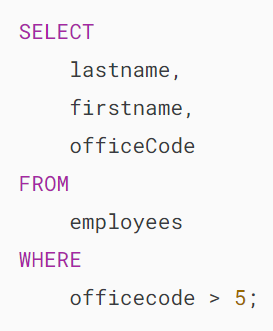


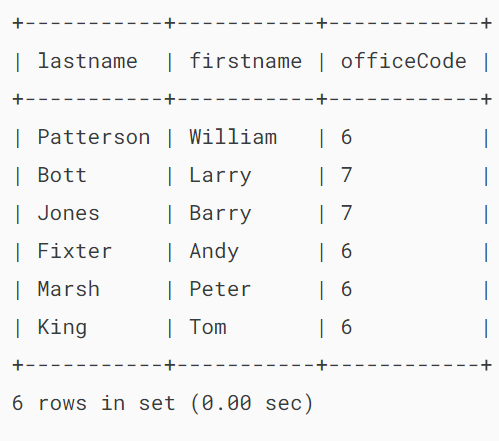
The following query uses the not equal to (<>) operator to find all employees who are not the Sales Rep:



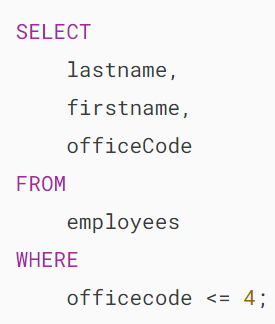


The following query finds employees whose office code is greater than 5:





The following query returns employees with office code less than or equal to 4 (<=4):



## 

## **Summary**

* Use the WHERE clause to filter rows by a condition.
* MySQL evaluates the WHERE clause after the FROM clause and before the SELECT and ORDER BY clauses.

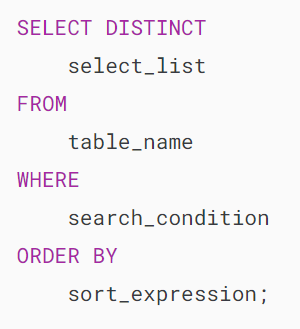
# **MySQL DISTINCT**

**Summary**: in this tutorial, you will learn how to use the MySQL DISTINCT clause in the SELECT statement to eliminate duplicate rows in a result set.

## **Introduction to MySQL DISTINCT clause**

When querying data from a table, you may get duplicate rows. To remove these duplicate rows, you use the DISTINCT clause in the [SELECT](https://www.mysqltutorial.org/mysql-basics/mysql-select-from/) statement.

Here’s the syntax of the DISTINCT clause:



In this syntax, you specify one or more columns that you want to select distinct values after the SELECT DISTINCT keywords.

If you specify one column, the DISTINCT clause will evaluate the uniqueness of rows based on the values of that column.

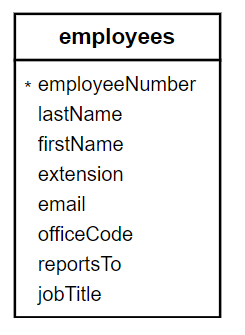
However, if you specify two or more columns, the DISTINCT clause will use the values of these columns to evaluate the uniqueness of the rows.

When executing the SELECT statement with the DISTINCT clause, MySQL evaluates the DISTINCT clause after the FROM, WHERE, and SELECT clause and before the ORDER BY clause:

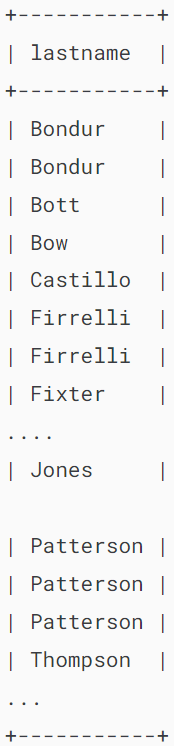
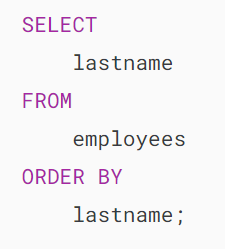


## **MySQL DISTINCT clause examples**

We’ll use the employees table from the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/):

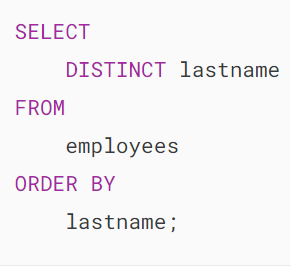


First, select the last names from the employees table using the following SELECT statement:



As shown clearly in the output, some employees have the same last names e.g.,Bondur,Firrelli .

Second, select unique last names by adding the DISTINCT clause like this:



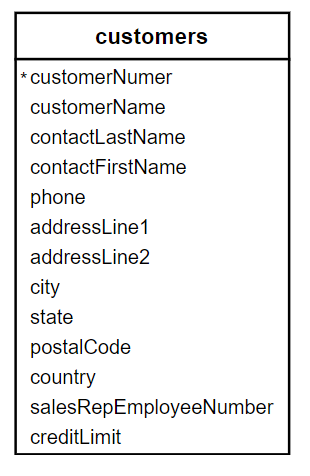
As you can see clearly from the output, the DISTINCT clause removes the duplicate last names from the result set.

## 

## **MySQL DISTINCT and NULL values**

When you specify a column that has [NULL](https://www.mysqltutorial.org/mysql-basics/mysql-null/) values in the DISTINCT clause, the DISTINCT clause will keep only one NULL value because it considers all NULL values are the same.

For example, the state column in the customers table has NULL values.



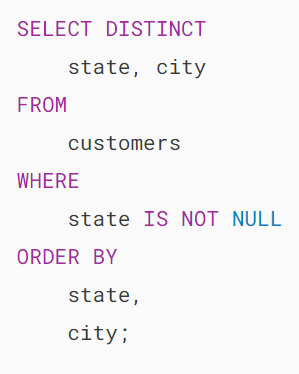
When you use the DISTINCT clause to query the states, you will see distinct states and NULL as follows:

## 

## **MySQL DISTINCT with multiple columns**

When you specify multiple columns in the DISTINCT clause, the DISTINCT clause will use the combination of values in these columns to determine the uniqueness of the row in the result set.

For example, to get a unique combination of city and state from the customers table, you use the following query:



Without the DISTINCT clause, you will get the duplicate combination of state and city as follows:

## 

## **Summary**

* Use the MySQL DISTINCT clause to remove duplicate rows from the result set returned by the SELECT clause.